Towards automated threshold selection for hydrocode simulation imagery

Patrick Kelly, Pat Fasel, John Hogden, James Howse, and Rich Fortson

CCS-3 Modeling, Algorithms, and Informatics
Los Alamos National Laboratory

May 23, 2006
LA-UR-06-3899

Abstract

We examine a methodology for automatically thresholding grayscale images. We begin by suggesting that this endeavor can benefit greatly from a clearly defined "goal". These goals are specific to a given problem, and are often stated in terms of shapes or other spatial characteristics of objects found in the image. The specific techniques that we consider attempt to analyze a criterion function defined in terms of connected-component features from the resultant monochrome images.

Introduction

In this paper, we discuss a class of automated thresholding techniques that make direct use of connected-component features to select an "optimal" threshold value. The specific features that are used will be customized using expert knowledge for the application at hand. The method we present has one primary advantage over other thresholding techniques – it can easily be compared to the way human experts select a threshold value interactively. As such, if our algorithm fails to work as desired on a specific set of data, it is relatively easy to incorporate additional expert knowledge in an effort to improve its performance. Additionally, the thresholding technique we describe can be used to summarize expert knowledge and convey it to future researchers.

We restrict our attention to the most common type of thresholding function, in which each pixel in the resultant monochrome image is derived solely from the single pixel value at the same location in the original grayscale image[1]. Any gray-level pixel with a value greater-than-or-equal-to a global threshold value, *T*, is replaced with *white*, and all other pixels are replaced with *black*. Figure 1 shows an image of the moon that has been thresholded with several different values for *T*.

[1] More complex thresholding functions also exist, in which a single pixel in the output monochrome image is derived from several pixels in the original grayscale image. Most of these techniques are commonly referred to as adaptive thresholding algorithms.

The ideal threshold value for a particular image is dependent on the ultimate "goal" that we are trying to achieve. For example, if we ask people to threshold our image of the moon (Figure 1) as a step toward determining its current phase, most people would choose a threshold value close to *T = 1*. We would be surprised if somebody chose a value *T = 192* to accomplish this goal. In contrast, if we ask people to choose a threshold that highlights the moon's surface morphology, some people may choose a threshold value near *T = 128*, while others may select threshold values closer to *T = 192*. This discrepancy might indicate that our goal – as stated – is too vague in its attempt to define what we desire unambiguously. Further refinement of our goal can decrease the subjectivity in choosing a threshold. Continuing with the moon example, if we ask people to find a threshold that highlights surface morphology by outlining the predominant valleys of the moon, we might very well achieve consensus that a threshold near *T = 128* is desired.

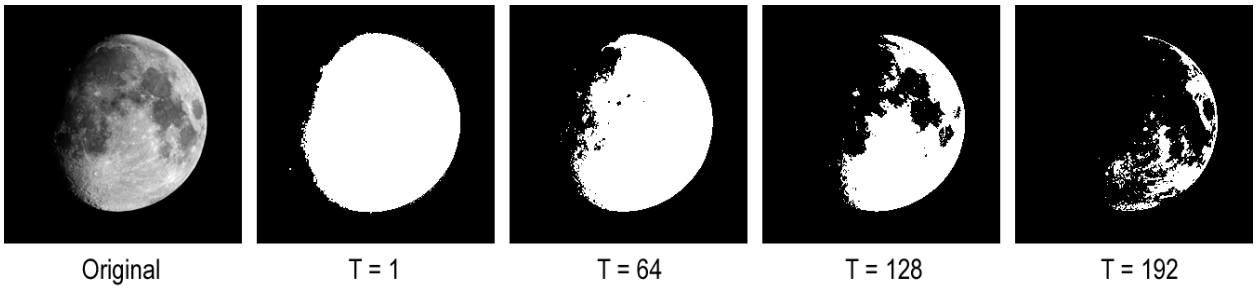

Figure 1. Thresholded images of the moon using various threshold values.

In a recent survey of image thresholding techniques, Sezgin and Sankur divide automated thresholding algorithms into six general categories [1]. The majority of these techniques identify a global threshold value using only gray-level histogram information. Some of these methods look for peaks and valleys in the histogram, while others model the histogram as "random processes generating foreground and background pixels", and still others attempt to select a threshold that optimizes a criterion such as "information transfer". These methods rely on assumptions about foreground and background pixels and the way they appear in the gray-level histogram. The advantages to these techniques include the fact that some of them are extremely fast, and memory requirements for computation are typically minimal. These histogram methods do not, however, make use of any spatial information at all. Since human experts are often interested in the shapes of objects, and shape information is not available in a brightness histogram, it is difficult to make direct use of expert knowledge within histogram-based techniques.

Another particularly interesting category of techniques that is mentioned in [1] includes methods that examine the spatial characteristics of white (or black) pixels in the resultant monochrome images. While most published algorithms in this category make use of attributes that are – in some sense – designed to be useful for any image, the general approach lends itself well to the development of specialized thresholding techniques for a given problem domain. In this sense, as long as analysts can describe what attributes they are ultimately looking for, an algorithm can be developed that specifically attempts to address their preferences. Thresholding techniques of this type are the focus of this paper.

Threshold Selection Using Connected-Component Features

Consider the example image shown in Figure 2. This image depicts four physical items (blocks) that lie within the image frame. If we threshold this image in order to perform image analysis, we would probably like to see four distinct objects with smooth boundaries clearly delineated in the resultant monochrome image. Thresholding techniques that work only through examination of the gray-level histogram, however, do not typically produce this result. Some of these techniques will isolate a single block (the brightest one), while others tend to produce an image with "partial" blocks in the image. Our goal of obtaining four distinct blocks can, however, be easily achieved interactively, where the user could select a threshold value near $T = 57$. As we look at smaller or larger threshold values, we begin to move away from this "optimal" result (see Figure 3).

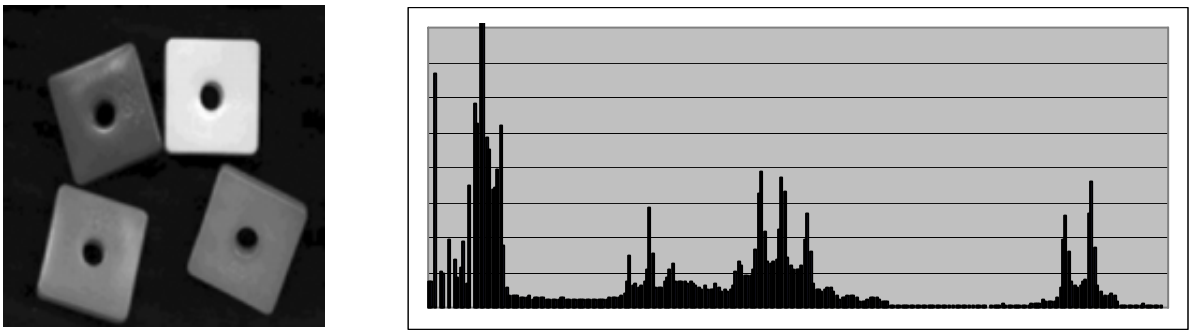

Figure 2. An example image with its gray-level histogram. Many histogram-based thresholding techniques will not produce an image where the four blocks are clearly delineated.

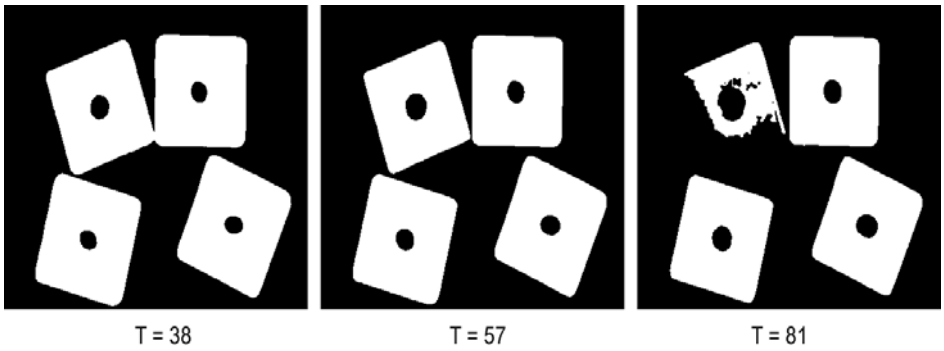

Figure 3. Interactive selection of a threshold for the "block" image. A threshold value of 57 results in four clearly identifiable blocks in the image. Smaller threshold values cause the uppermost blocks to "touch", while larger thresholds cause some of the blocks to "deteriorate".

In this example, we can define what we would like to achieve in a fairly straightforward manner – "*We would like to threshold this image in such a way as to see four distinct white objects*". In practice, we may find that this specification is not sufficient. A large number of possible threshold values will yield "*four distinct white objects*" in the resultant monochrome image, but most of these images do not depict the four separate "blocks" that we are ultimately interested in. Examples of this can be seen in Figure 4. A threshold value of $T = 4$ results in an image that contains three extremely small objects (each consisting of only a few pixels) and one very large object that covers nearly the entire image. A threshold value of $T = 26$ will also produce four distinct white objects, but one of these objects actually represents two blocks that are "touching" in the image, with another object simply being a small piece of "speckle". Something akin to our desired result can be obtained with $T = 55$.

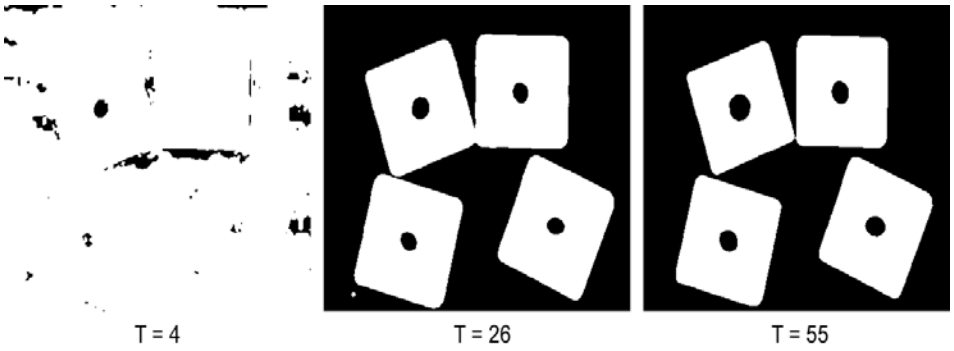

Figure 4. Designing an automated algorithm for thresholding the "block" image. Several different thresholds will produce results that depict four distinct white objects in the thresholded image, although some of these will not reflect what we were ultimately seeking.

If we can define a criterion function that is maximized when our output monochrome image depicts the four "blocks" as distinct, separate white objects in the image, then we can use this function to automatically choose a threshold value that will accomplish our desired goal. We start with the following definition for our criterion, which is a function of the input monochrome image at threshold T:

$$f(\mathrm{I_T}) = \begin{cases} \sum_{j=1}^{4} \#\text{pixels}_j & \text{if } \#\text{objects} = 4 \\ 0 & \textit{otherwise} \end{cases}$$

The plot of this function for our "block" image example is shown in Figure 5. Every threshold that produces a monochrome image with exactly four distinct white objects (e.g. four connected components) will produce a non-zero value of this criterion function. The values on this plot are simply the total number of pixels contained in the four white objects. If we choose the threshold value *T* that maximizes this function, our result will be *T = 4*, which is obviously not the answer we desire.

Our remedy for this situation can be a simple bit of preprocessing built in to our criterion. Specifically, if our function ignores very large objects (such as those that touch one of the image borders) as well as very small objects (such as those containing fewer that 100 pixels), then it will successfully select *T = 55* as our desired threshold (see Figure 6).

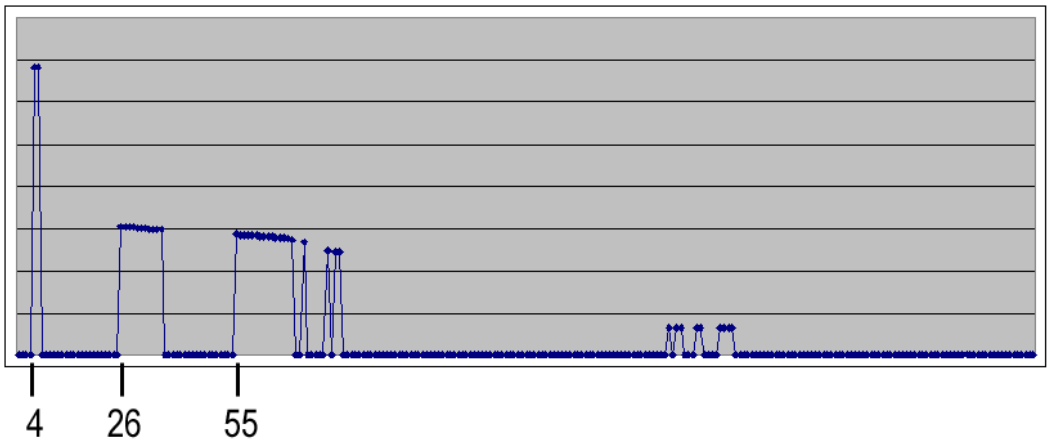

Figure 5. Criterion function values as our threshold varies from 0 to 255. Valid points (non-zero) on this plot exist for thresholds where four distinct white objects are visible in the "block" image. Thresholds that yield some other number of objects produce a criterion value of zero.

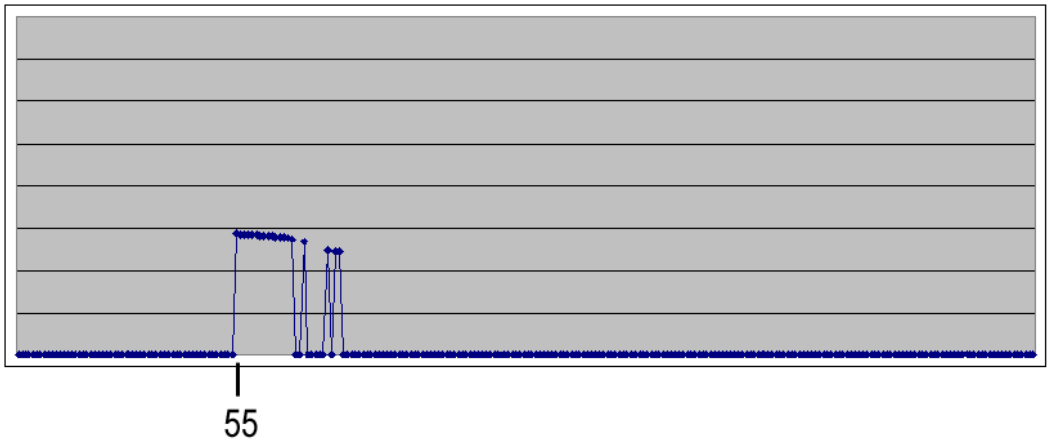

Figure 6. Criterion function values when very small/large objects are ignored in the image.

This same methodology can be used to incorporate expert knowledge into a thresholding routine for almost any application. Generally speaking, the criterion function should be specialized enough to "get the job done", while not being overly restrictive (which might cause it to be "too specific" and, as a result, fail when presented with new data).

Automated Thresholding of Hydrocode Simulation Data

We are currently studying images that are generated through hydrocode simulations of a supersonic plasma jet interacting with a shock wave [2]. As with the moon example above, when we consider the problem of analyzing hydrocode simulation data, different threshold values may result in images that capture distinctly different aspects of the underlying data. An example of this can be seen in Figure 7, where different threshold values have been used to generate monochrome imagery depicting the "mushroom jet" and the "bow shock" for a specific frame in a simulation.

Figure 7. Applying thresholds to hydrocode simulation data.

We plan to use thresholded imagery to automatically compare simulated images with experimental images that are captured in the laboratory. It has been suggested that the relative positions of the jet and the bow shock are important for characterizing these images [3], so our goal is to find two thresholds for each image — one threshold that can be used to extract the bow shock and another to extract the jet. For purposes of this paper, our attention is restricted to finding threshold values that result in monochrome images depicting bow shocks in the simulation.

Using the principles described in the preceding section of this paper, we tried to address to the specific problem of thresholding hydrocode simulation data by asking ourselves a simple question – "What do we look for when we threshold this image by hand?" Our answer was then used to build an automated algorithm that would mimic our interactive approach to choosing a threshold value. It was important in this exercise to operationally define what we were looking at in terms of the image. Even if our automated algorithm ultimately fails in its attempt to characterize what human experts would do, it still provides a starting point that can be refined.

The entire set of data we had available consisted of 61 separate images taken at time steps of 30ns – 90ns for the simulation. The two "bow shocks" from the upper and lower halves of the simulation appear to come together around 46ns. For time steps before 46ns, we wanted our thresholded images to clearly depict the bow shocks as smooth curves that were not yet "touching" in the data. For time steps after 46ns, we wanted to clearly show that the bow shocks had come together (see Figure 8).

Our observation was that – when thresholding these images by hand – the images exhibited specific properties when we were near the desired threshold. As we moved our threshold from a large value near 255 (where the image is nearly all-black) to a small value near 0 (where the image is nearly all-white), there was always some point at which one of two things happened. In one case, the white region in the middle of the image would suddenly connect all the way from the left-hand edge to the right-hand edge. This phenomenon would occur in images where the bow shocks were still distinctly separated. In the second case, black areas would begin to "fracture" in some way (typically with the appearance of small white holes where the density inside of the bow shock was not distinguishable from the density outside of the bow shock). This would occur in images where the bow shocks had already come together (see Figure 9). Using this knowledge, we were able to build an automated threshold routine that can successfully find bow shocks in this set of simulated data.

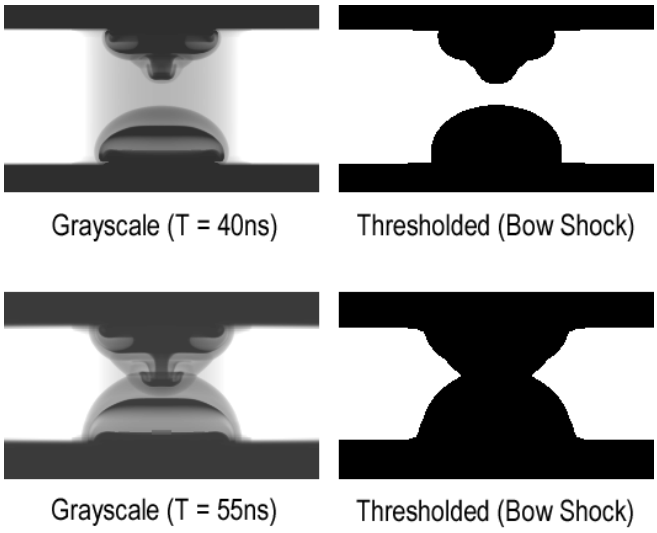

Figure 8. Selecting threshold values for hydrocode simulation data in an effort to generate monochrome images that depict "bow shocks" for analysis.

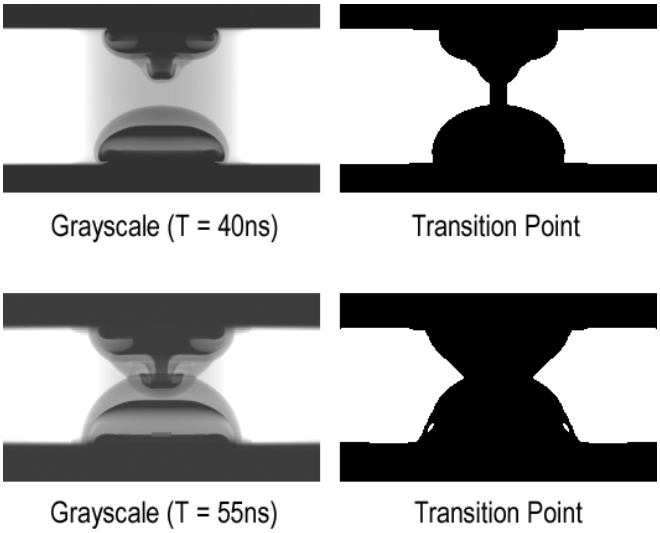

Figure 9. Threshold values just past the "transition point" for the case where (A) the simulated bow shocks had not yet come together; and (B) where the simulated bow shocks had already merged

Conclusions

We studied a class of automatic image thresholding procedures that choose a threshold based on features derived from connected-components in the resultant monochrome images. We showed that it is relatively easy to build expert knowledge into these types of thresholding techniques. This methodology was applied to the problem of thresholding our current set of hydrocode simulation data, and we demonstrated that, for this particular set of data, we could successfully threshold the data automatically in such a way as to depict they data "bow shocks". We hope to extend these techniques to the problem of finding bow shocks in similar data sets (produced by slightly different simulations), as well as to the problem of finding thresholds that capture other significant features of the images (e.g., laser driven jets).

Bilbiography

[1] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging* 13(1), 146-165, January 2004.

[2] J. M. Foster, B. H. Wilde, P. A. Rosen, T. S. Perry, M. Fell, M. J. Edwards, B. F. Lasinski, R. E. Turner, and M. L. Gittings. Supersonic jet and shock interactions, Physics of Plasmas, 9(5), 2251-2263, May 2002.

[3] B. H. Wilde. Personal communication.